ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

Wenpeng Yin, Hinrich Schütze
Center for Information and Language Processing
University of Munich, Germany
wenpeng@cis.lmu.de

Bing Xiang, Bowen Zhou
IBM Watson
Yorktown Heights, NY, USA
bingxia,zhou@us.ibm.com

Abstract

How to model a pair of sentences is a critical issue in many natural language processing (NLP) tasks such as answer selection (AS), paraphrase identification (PI) and textual entailment (TE). Most prior work (i) deals with one individual task by fine-tuning a specific system; (ii) models each sentence separately, without considering the impact of the other sentence; or (iii) relies fully on manually designed, task-specific linguistic features. This work presents a general **A**ttention **B**ased **C**onvolutional **N**eural **N**etwork (ABCNN) for modeling a pair of sentences. We make three contributions. (i) ABCNN can be applied to a wide variety of tasks that require modeling of sentence pairs. (ii) We propose three attention schemes that integrate mutual influence between sentences into CNN; thus, the representation of each sentence takes into consideration its counterpart. These interdependent sentence pair representations are more powerful than isolated sentence representations. (iii) ABCNN achieves state-of-the-art performance on AS, PI and TE tasks.

1 Introduction

How to model a pair of sentences is a critical issue in many NLP tasks such as answer selection (AS) (Yu et al., 2014; Feng et al., 2015), paraphrase identification (PI) (Madnani et al., 2012; Yin and Schütze, 2015a), textual entailment (TE) (Marelli et al., 2014a; Bowman et al., 2015a) and machine translation (Bahdanau et al., 2015).

Most prior work models each sentence separately, without considering the impact of the other sentence. This neglects the mutual influence of the two sentences in the context of the task. It also contradicts what humans do when comparing two sentences. We usually focus on key parts of one sentence by extracting parts from the other sentence that are related by identity, synomymy, antonymy and other relations. Thus, human beings model the two sentences together, using the content of one sentence to guide the representation of the other.

AS	$\mathbf{s_0}$	how much did Waterboy *gross*?
	$\mathbf{s}_1^+$	the movie *earned* $161.5 million
	$\mathbf{s}_1^-$	this was Jerry Reeds final film appearance
PI	$\mathbf{s_0}$	she struck a deal with RH to pen a book *today*
	$\mathbf{s}_1^+$	she signed a contract with RH to write a book
	$\mathbf{s}_1^-$	she denied *today* that she struck a deal with RH
TE	$\mathbf{s_0}$	an ice skating rink placed *outdoors* is *full of people*
	$\mathbf{s}_1^+$	a *lot of people* are in an ice skating park
	$\mathbf{s}_1^-$	an ice skating rink placed *indoors* is *full of people*

Figure 1: Positive ($< s_0, s_1^+ >$) and negative ($< s_0, s_1^- >$) examples for AS, PI and TE tasks. RH = Random House

Figure 1 demonstrates that each sentence of a pair partially determines which parts of the other sentence we should focus on. For AS, correctly answering s_0 requires putting attention on "gross": s_1^+ contains a corresponding unit ("earned") while s_1^- does not. For PI, focus should be removed from "today" to correctly recognize ($< s_0, s_1^+ >$) as paraphrases and ($< s_0, s_1^- >$) as non-paraphrases. For TE, we need to focus on "full of people" (to recognize TE for $< s_0, s_1^+ >$) and on "outdoors" / "indoors" (to recognize non-TE for $< s_0, s_1^- >$). These examples show the need for an architecture that computes different representations of s_i for different s_{i-1}'s

($i \in \{0, 1\}$).

In this paper, we present such an architecture: ABCNN, an attention-based convolutional neural network that has a powerful mechanism for modeling a sentence pair by taking into account the interdependence between the two sentences. ABCNN is a general architecture that can handle a wide variety of sentence pair modeling tasks.

Some prior work proposes simple mechanisms that can be interpreted as controlling varying attention; e.g., Yih et al. (2013) employ word alignment to match related parts of the two sentences. In contrast, our attention scheme based on CNN is able to model relatedness between two parts fully automatically. Moreover, attention at multiple levels of granularity, not only at the word level, is achieved as we stack multiple convolution layers that increase abstraction. As far as we know, this is the first NLP paper that incorporates attention into CNNs.

Section 2 discusses related work. Section 3 introduces BCNN, a network that models two sentences in parallel with shared weights, but without attention. Section 4 presents three different attention mechanisms and their realization in ABCNN, an architecture that is based on BCNN. Section 5 evaluates the models on AS, PI and TE tasks.

2 Related Work

There has been a lot of neural network research on modeling sentence pairs for AS, PI and TE. For AS, Yu et al. (2014) present a bigram CNN to model question and answer candidates. Yang et al. (2015) extend this method and get state-of-the-art performance on the WikiQA dataset (Section 5.2). Feng et al. (2015) test various setups of a bi-CNN architecture on an insurance domain QA dataset. Tan et al. (2015) explore bidirectional long short-term memory (LSTM, Hochreiter and Schmidhuber (1997)) on the same dataset. Our approach is different because we do not model the sentences by two independent neural networks in parallel, but instead as an interdependent sentence pair, using attention.

For PI, Blacoe and Lapata (2012) form sentence representations by summing up word embeddings. Socher et al. (2011) use recursive autoencoder (RAE) to model representations of local phrases in sentences, then pool similarity values of phrases from the two sentences as features for binary classification. Yin and Schütze (2015a) present a similar model in which RAE is replaced by CNN. In all three papers, the representation of each sentence is not influenced by the other's – in contrast to our attention-based model.

For TE, Bowman et al. (2015b) employ recursive neural networks to encode entailment on SICK (Marelli et al., 2014b). Rocktäschel et al. (2015) present an attention-based LSTM for the Stanford natural language inference corpus (Bowman et al., 2015a). Our system is the first CNN-based work on TE.

Some prior work aims to solve a general sentence matching problem. Hu et al. (2014) present two CNN architectures, ARC-I and ARC-II, for sentence matching. ARC-I focuses on sentence representation learning while ARC-II focuses on matching features on phrase level. Both systems were tested on PI, sentence completion (SC), tweet-response matching. Yin and Schütze (2015b) propose the MultiGranCNN architecture to model general sentence matching based on phrase matching on multiple levels of granularity and get promising results for PI and SC. Wan et al. (2015) try to match two sentences in AS and SC by multiple sentence representations, each coming from the local representations of two LSTMs. Our work is the first one to investigate attention for the general sentence matching task.

3 BCNN: Basic Bi-CNN

We now introduce our basic (non-attention) CNN that is based on Siamese architecture, i.e., it consists of two weight-sharing CNNs, each processing one of the two sentences, and a final layer that solves the sentence pair task. See Figure 2. We refer to this architecture as *BCNN*. The next section will then introduce an attention architecture that extends BCNN. Table 1 gives our notational conventions.

In our implementation and also in the mathematical formalization of the model given below, we pad the two sentences to have the same length $s = \max(s_0, s_1)$. However, in the figures we show different lengths because this gives a better intuition of how the model works.

BCNN has four types of layers.

symbol	description
s, s_0, s_1	sentence or sentence length
v	word
w	filter width
d_i	dimensionality of input to layer $i+1$
$\mathbf{W}$	weight matrix

Table 1: Notation

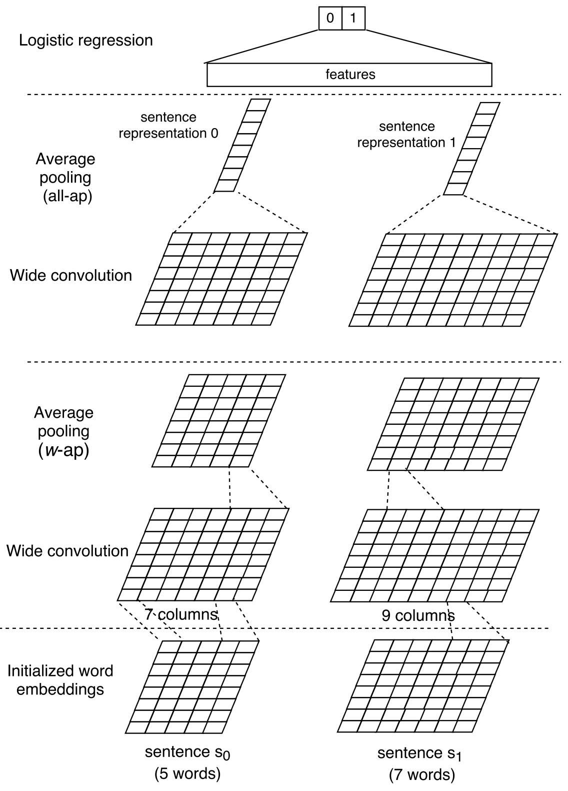


Figure 2: BCNN: ABCNN without Attention

Input layer. In the example in the figure, the two input sentences have 5 and 7 words, respectively. Each word is represented as a d_0-dimensional precomputed word2vec (Mikolov et al., 2013) embedding,[1] $d_0 = 300$. As a result, each sentence is represented as a feature map of dimension $d_0 \times s$.

Convolution layer. Let $v_1, v_2, \ldots, v_s$ be the words of a sentence and $\mathbf{c}_i \in \mathbb{R}^{wd_0}$, $0 < i < s + w$, the concatenated embeddings of $v_{i-w+1}, \ldots, v_i$ where embeddings for v_i, $i < 1$ and $i > s$, are set to zero. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d_1}$ for the *phrase* $v_{i-w+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d_1 \times wd_0}$ as follows:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \quad (1)$$

[1]https://code.google.com/p/word2vec/

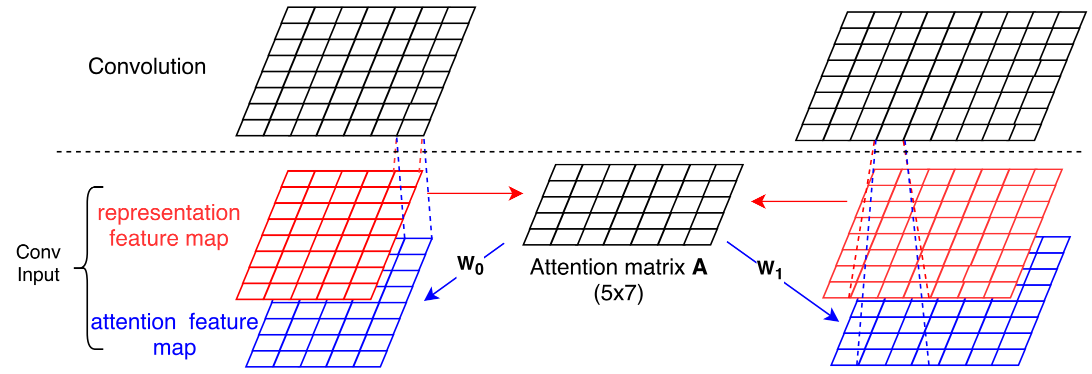
where $\mathbf{b} \in \mathbb{R}^{d_1}$ is the bias. We use *wide convolution*; i.e., we apply the convolution weights $\mathbf{W}$ to words v_i, $i < 1$ and $i > s$, because this makes sure that each word v_i, $1 \leq i \leq s$, can be detected by all weights in $\mathbf{W}$ – as opposed to only the rightmost (resp. leftmost) weights for initial (resp. final) words in narrow convolution.

Average pooling layer. Pooling (max, min, average etc) is commonly used to extract robust features from convolution. In this paper, we introduce attention weighting as an alternative, but use average pooling as a baseline as follows. For the output feature map of the last convolution layer, we do column-wise averaging over *all columns*, denoted as *all-ap*. This will generate a representation vector for each of the two sentences, shown as the top "Average pooling (*all-ap*)" layer below "Logistic regression" in Figure 2. These two representations are then the basis for the sentence pair decision.

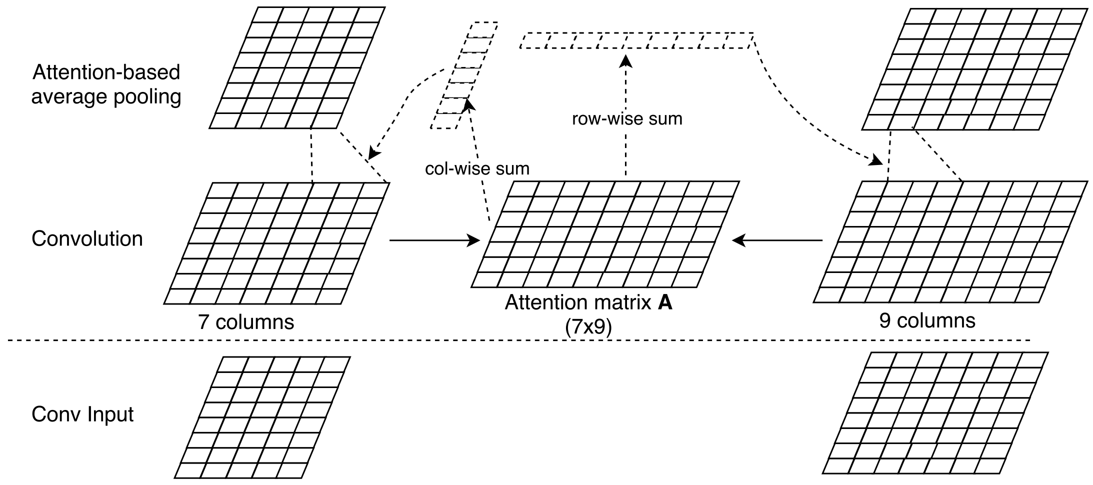
For the output feature map of non-final convolution layers, we do column-wise averaging over *windows of w consecutive columns*, denoted as *w-ap*; shown as the lower "Average pooling (*w-ap*)" layer in Figure 2. For filter width w, a convolution layer transforms an input feature map of s columns into a new feature map of $s + w - 1$ columns; average pooling transforms this back to s columns. This architecture supports stacking an arbitrary number of convolution-pooling blocks to extract increasingly abstract features. Input features to the bottom layer are words, input features to the next layer are short phrases and so on. Each level generates more abstract features of higher granularity.

Output layer. The last layer is an output layer, chosen according to the task; e.g., for binary classification tasks, this layer is logistic regression (see Figure 2). Other output layers are introduced below.

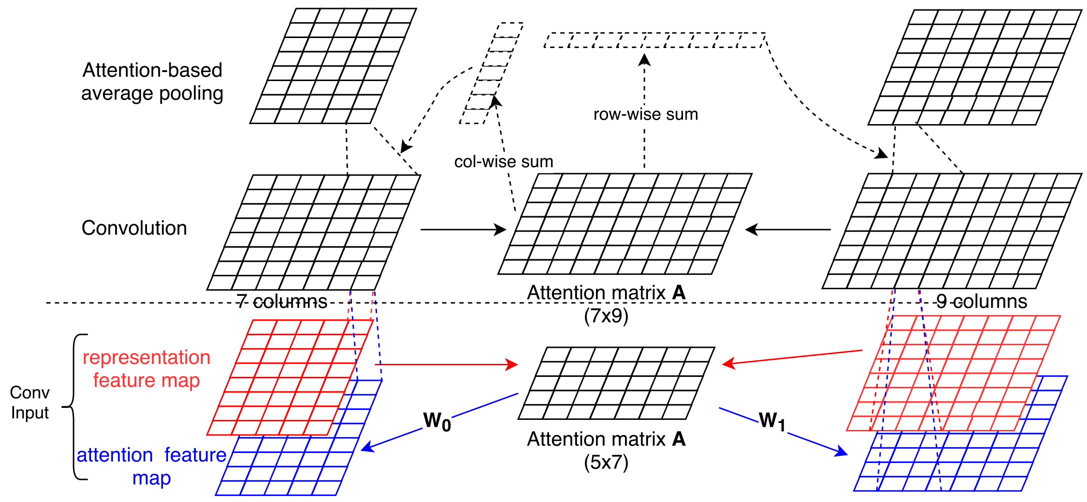
We found that in most cases, performance is boosted if we provide the output of *all pooling layers* as input to the output layer. For each non-final average pooling layer, we perform *w-ap* (pooling over windows of w columns) as described above, but we also perform *all-ap* (pooling over all columns) and forward the result to the output layer. This improves performance because representations from different layers cover the properties of the sentences at different levels of abstraction and all of these levels can be important for a particular sentence pair.



(a) One Block in ABCNN-1



(b) One Block in ABCNN-2



(c) One Block in ABCNN-3

Figure 3: Three ABCNN architectures

4 ABCNN: Attention-Based BCNN

We introduce three attention mechanisms for modeling sentence pairs into BCNN; see Figure 3.

4.1 ABCNN-1

ABCNN-1 employs an attention feature matrix $\mathbf{A}$ to influence convolution. Attention features are intended to weight those units of s_i more highly in convolution that are relevant to a unit of s_{1-i} ($i \in \{0,1\}$); we use the term "unit" here to refer to words on the lowest level and to phrases on higher levels of the network. Figure 3(a) shows two *unit representation feature maps* in red: this part of ABCNN-1 is the same as in BCNN (see Figure 2). Each column is the representation of a unit, a word on the lowest level and a phrase on higher levels. We first describe the attention feature matrix $\mathbf{A}$ informally (layer "Conv input", middle column, in Figure 3(a)). $\mathbf{A}$ is generated by matching units of the left matrix with units of the right matrix such that the attention values of row i in $\mathbf{A}$ denote the attention distribution of the i-th unit of s_0 with respect to s_1 and the attention values of column j in $\mathbf{A}$ denote the attention distribution of the j-th unit of s_1 with respect to s_0. $\mathbf{A}$ can be viewed as a new feature map of s_0 (resp. s_1) in row (resp. column) direction because each row (resp. column) is a new feature vector of a unit in s_0 (resp. s_1). Thus, it makes sense to combine this new feature map with the representation feature maps and use both as input to convolution. We achieve this by transforming $\mathbf{A}$ into the two blue matrices in Figure 3(a) that have the same format as the representation feature maps. As a result, the new input of convolution has two feature maps for each sentence (shown in red and blue). Our motivation is that the attention feature map will guide the convolution to learn "counterpart-biased" sentence representations.

More formally, let $\mathbf{F}_{i,r} \in \mathbf{R}^{d\times s}$ be the representation feature map of sentence i ($i \in \{0,1\}$). Then we define the attention matrix $\mathbf{A} \in \mathbf{R}^{s\times s}$ as follows:

$$\mathbf{A}_{i,j} = \text{match-score}(\mathbf{F}_{0,r}[:,i], \mathbf{F}_{1,r}[:,j]) \quad (2)$$

The function match-score can be defined in a variety of ways. We found that $1/(1+|x-y|)$ works well where $|\cdot|$ is Euclidean distance.

Given attention matrix $\mathbf{A}$, we generate the *attention feature map* $\mathbf{F}_{i,a}$ for s_i as follows:

$$\begin{aligned} \mathbf{F}_{0,a} &= \mathbf{W}_0 \cdot \mathbf{A}^\intercal & (3) \\ \mathbf{F}_{1,a} &= \mathbf{W}_1 \cdot \mathbf{A} & (4) \end{aligned}$$

The weight matrices $\mathbf{W}_0 \in \mathbf{R}^{d\times s}$, $\mathbf{W}_1 \in \mathbf{R}^{d\times s}$ are parameters of the model to be learned in training.[2]

We stack the representation feature map $\mathbf{F}_{i,r}$ and the attention feature map $\mathbf{F}_{i,a}$ as an order 3 tensor and feed it into convolution to generate a higher-level representation feature map for s_i ($i \in \{0,1\}$). In Figure 3(a), s_0 has has 5 units, s_1 has 7. The output of convolution (shown in the top layer, filter width $w=3$) is a higher-level representation feature map with 7 columns for s_0 and 9 columns for s_1.

4.2 ABCNN-2

ABCNN-1 computes attention weights *directly on the representation* with the aim of *improving the features computed by convolution.* ABCNN-2 instead computes attention weights *on the output of convolution* with the aim of *reweighting this convolution output.* In the example shown in Figure 3(b), the feature maps output by convolution for s_0 and s_1 have 7 and 9 columns, respectively; each column is the representation of a unit. The attention matrix $\mathbf{A}$ compares all units in s_0 with all units of s_1. We sum all attention values for a unit to derive a single attention weight for that unit. This corresponds to summing all values in a row of $\mathbf{A}$ for s_0 (resulting in the column vector of size 7 shown) and summing all values in a column for s_1 (resulting in the row vector of size 9 shown).

More formally, let $\mathbf{A} \in \mathbf{R}^{s\times s}$ be the attention matrix, $a_{0,j} = \sum \mathbf{A}[j,:]$ the attention weight of unit j in s_0, $a_{1,j} = \sum \mathbf{A}[:,j]$ the attention weight of unit j in s_1 and $\mathbf{F}^c_{i,r} \in \mathbf{R}^{d\times(s_i+w-1)}$ the output of convolution for s_i. Then the j-th column of the new feature map $\mathbf{F}^p_{i,r}$ generated by *w-ap* is derived by:

$$\mathbf{F}^p_{i,r}[:,j] = \sum_{k=j:j+w} a_{i,k} \cdot \mathbf{F}^c_{i,r}[:,k], \quad j = 1\ldots s_i$$

Notice that $\mathbf{F}^p_{i,r} \in \mathbf{R}^{d\times s_i}$, i.e., ABCNN-2 pooling generates an output feature map of the same size as

[2]The weights of the two matrices are shared in our implementation to reduce the number of parameters of the model.

the input feature map of convolution. This allows us to stack multiple convolution-pooling blocks to extract features of increasing abstraction.

There are three main differences between ABCNN-1 and ABCNN-2. (i) Attention in ABCNN-1 impacts *convolution indirectly* while attention in ABCNN-2 influences *pooling* through *direct* attention weighting. (ii) ABCNN-1 requires the two matrices $\mathbf{W}_i$ to convert the attention matrix into attention feature maps; and the input to convolution has two times as many feature maps. Thus, ABCNN-1 has more paramaters than ABCNN-2 and is more vulnerable to overfitting. (iii) As pooling appears after convolution, pooling handles larger-granularity units than convolution; e.g., if the input to convolution has word level granularity, then the input to pooling has phrase level granularity, the phrase size being equal to filter size w. Thus, ABCNN-1 and ABCNN-2 implement attention mechanisms for linguistic units of different granularity. This is exactly the motivation for the third ABCNN architecture, ABCNN-3.

4.3 ABCNN-3

ABCNN-3 combines ABCNN-1 and ABCNN-2 by stacking them. See Figure 3(c). ABCNN-3 combines the strengths of ABCNN-1 and ABCNN-2 by allowing the attention mechanism to operate (i) on both the convolution and pooling parts of a convolution-pooling block and (ii) on both the input granularity and the more abstract output granularity.

5 Experiments

We test the proposed architectures on three tasks: answer selection, paraphrase identification and textual entailment.

5.1 Common Training Setup

For all tasks, words are initialized by 300-dimensional word2vec embeddings and not changed during training. A single randomly initialized embedding[3] is created for all unknown words by uniform sampling from [-.01,.01]. We employ Adagrad (Duchi et al., 2011) and L_2 regularization.

Network configuration. Each network in the experiments below consists of (i) an initialization block b_1 that initializes words by word2vec embeddings, (ii) a stack of $k-1$ convolution-pooling blocks $b_2, \ldots, b_k$, computing increasingly abstract features, and (ii) one final *LR layer* (logistic regression layer) as shown in Figure 2.

The input to the LR layer consists of kn features – each block provides n similarity scores (such as cosine similarity). Figure 2 shows the two sentence vectors output by the final block b_k of the stack; this is the basis of the last n similarity scores. As we explained in the final paragraph of Section 3, we perform *all-ap* pooling for *all blocks*, not just for b_k. Thus we get one sentence representation each for s_0 and s_1 for each block $b_1, \cdots, b_k$. We compute n similarity scores for each block (based on the two sentence representations) and forward these kn scores as input to the LR layer.

Depending on the task, we use different methods for computing the similarity score: see below.

Layerwise training. In our training regime, we first train a network consisting of just one conv-pooling block b_2. We then create a two-block network, initialize its b_2 block with the corresponding one-block parameters and train b_3 keeping the previously learned weights for b_2 fixed. We repeat this procedure until all $k-1$ conv-pooling blocks are trained. We found that this training regime gives us good performance and shortens training times considerably. Since similarity scores of lower blocks are kept unchanged once they have been learned, this also has the nice effect that "simple" similarity scores (those based on surface features) are learned first and subsequent training phases can focus on complementary scores derived from more complex abstract features.

Classifier. We found that performance increases if we do not use the output of the LR layer as the final decision, but instead train linear SVM or logistic regression with default parameters[4] directly on the input to the LR layer (i.e., on the input that is generated by the k-block stack after network training is completed). Direct training of SVMs/LR seems to get closer to the global optimum than gradient descent training of CNNs.

Table 2 shows the values of the hyperparameters. Hyperparameters were tuned on dev.

[3]This worked better than discarding unknown words.

[4] `http://scikit-learn.org/stable/` for both.

	#CL	AS			PI			TE		
		lr	w	L_2	lr	w	L_2	lr	w	L_2
ABCNN-1	1	.08	4	.0004	.08	3	.0002	.08	3	.0006
ABCNN-1	2	.085	4	.0006	.085	3	.0003	.085	3	.0006
ABCNN-2	1	.05	4	.0003	.085	3	.0001	.09	3	.00065
ABCNN-2	2	.06	4	.0006	.085	3	.0001	.085	3	.0007
ABCNN-3	1	.05	4	.0003	.05	3	.0003	.09	3	.0007
ABCNN-3	2	.06	4	.0006	.055	3	.0005	.09	3	.0007

Table 2: Hyperparameters. lr: learning rate. #CL: number convolution layers. w: filter width. The number of convolution kernels d_i ($i > 0$) is 50 throughout.

5.2 Answer Selection

We use WikiQA,[5] an open domain question-answer dataset. We use the subtask that assumes that there is at least one correct answer for a question. The corresponding dataset consists of 20,360 question-candidate pairs in train, 1,130 pairs in dev and 2,352 pairs in test where we adopt the standard setup of only considering questions that have correct answers for evaluation. Following Yang et al. (2015), we truncate answers to 40 tokens.

The task is to rank the candidate answers based on their relatedness to the question. Evaluation measures are mean average precision (MAP) and mean reciprocal rank (MRR).

We compare with the seven **baselines** considered by Yang et al. (2015): (i) WordCnt: count the number of non-stopwords in the question that also occur in the answer; (ii) WgtWordCnt: reweight the counts by the IDF values of the question words; (iii) LCLR (Yih et al., 2013) makes use of rich lexical semantic features, including word/lemma matching, WordNet (Miller, 1995) and distributional models; (iv) PV: Paragraph Vector (Le and Mikolov, 2014); (v) CNN: bigram convolutional neural network (Yu et al., 2014); (vi) PV-Cnt: combine PV with (i) and (ii); (vii) CNN-Cnt: combine CNN with (i) and (ii).

5.2.1 Task-Specific Setup

We use cosine similarity as the similarity score for this task. In addition, we use sentence lengths, WordCnt and WgtWordCnt. Thus, the final input to the LR layer has size $k + 4$: one cosine for each of the k blocks and the four additional features.

[5]`http://aka.ms/WikiQA` (Yang et al., 2015)

method		MAP	MRR
Baselines	WordCnt	0.4891	0.4924
	WgtWordCnt	0.5099	0.5132
	LCLR	0.5993	0.6086
	PV	0.5110	0.5160
	CNN	0.6190	0.6281
	PV-Cnt	0.5976	0.6058
	CNN-Cnt	<u>0.6520</u>	<u>0.6652</u>
BCNN	one-conv	0.6629	0.6813
	two-conv	0.6593	0.6738
ABCNN-1	one-conv	0.6810	0.6979
	two-conv	0.6855	0.7023
ABCNN-2	one-conv	0.6885	0.7054
	two-conv	0.6879	0.7068
ABCNN-3	one-conv	0.6914	**0.7127**
	two-conv	**0.6921**	0.7108

Table 3: Results on WikiQA. State-of-the-art baselines are underlined. Best result per column is bold.

5.2.2 Results

Table 3 shows performance of the baselines, of BCNN and of the three ABCNN architectures. For CNNs, we test one (one-conv) and two (two-conv) convolution-pooling blocks.

The non-attention network BCNN already performs better than the baselines. If we add attention mechanisms, then the performance further improves by several points. Comparing ABCNN-2 with ABCNN-1, we find ABCNN-2 is slightly better even though ABCNN-2 is the simpler architecture. If we combine ABCNN-1 and ABCNN-2 to form ABCNN-3, we get further improvement.[6]

This can be explained by ABCNN-3's ability to take attention of more fine-grained granularity into consideration in each convolution-pooling block while ABCNN-1 and ABCNN-2 consider attention only at convolution input or only at pooling input, respectively. We also find that stacking two convolution-pooling blocks does not bring consistent improvement and therefore do not test deeper architectures.

In summary, the attention mechanism performs better by a large margin on answer selection than previous work that does not use attention. This is evidence that attention is useful for this task.

[6] If we limit the input to LR layer to the k similarity scores in ABCNN-3 (two conv), results are .660 (MAP) / .677 (MRR).

5.3 Paraphrase Identification

We use Microsoft Research Paraphrase (MSRP) corpus (Dolan et al., 2004). The training set contains 2753 true / 1323 false and the test set 1147 true / 578 false paraphrase pairs. We randomly select 400 pairs from train and use them as dev set; but we still report results for training on the entire training set. For each triple (label, s_0, s_1) in train, we also add (label, s_1, s_0) to train to make best use of the training data. Systems are evaluated by accuracy and F_1.

We compare our system with three representative neural network (NN) **baselines**: (i) RAE (Socher et al., 2011), recursive autoencoder based PI; (ii) Bi-CNN-MI (Yin and Schütze, 2015a), a bi-CNN architecture, (iii) MPSSM-CNN (He et al., 2015), the state-of-the-art NN system for PI; and two non-NN systems: (iv) MT (Madnani et al., 2012), a system that combines machine translation metrics;[7] (v) MF-TF-KLD (Ji and Eisenstein, 2013), the state-of-the-art non-NN system.

5.3.1 Task-Specific Setup

In this task, we add the 15 MT features from (Madnani et al., 2012) and the lengths of the two sentences. In addition, we compute ROUGE-1, ROUGE-2 and ROUGE-SU4 (Lin, 2004), which are scores measuring the match between the two sentences on (i) unigrams, (ii) bigrams and (iii) unigrams and skip-bigrams (maximum skip distance of four), respectively. In this task, we found transforming Euclidean distance into similarity score by $1/(1 + |x - y|)$ performs better than cosine similarity. Additionally, we use dynamic pooling (Yin and Schütze, 2015a) of the attention matrix and forward pooled values of all blocks to the LR layer. This gives us slightly better performance than only forwarding sentence-level matching features.

5.3.2 Results

Table 4 shows that BCNN is slightly worse than the state-of-the-art whereas ABCNN-1 roughly matches it. ABCNN-2 is slightly above the state-of-the-art. ABCNN-3 outperforms the state-of-the-art clearly in accuracy and F_1.[8] Two convolution layers only bring small improvements over one.

method		acc	F_1
Baselines	majority voting	66.5	79.9
	RAE	76.8	83.6
	Bi-CNN-MI	78.4	84.6
	MPSSM-CNN	<u>78.6</u>	<u>84.7</u>
	MT	76.8	83.8
	MF-TF-KLD	<u>78.6</u>	84.6
BCNN	one-conv	78.1	84.1
	two-conv	78.3	84.3
ABCNN-1	one-conv	78.5	84.5
	two-conv	78.5	84.6
ABCNN-2	one-conv	78.6	84.7
	two-conv	78.8	84.7
ABCNN-3	one-conv	78.8	**84.8**
	two-conv	**78.9***	**84.8**

Table 4: Results on MSRP. Significant improvement over state-of-the-art is marked with $*$ (McNemar's Chi-squared test, $p < .05$).

	ORIG	NONOVER
0	children in red shirts are playing in the leaves	children red shirts playing
	three kids are sitting in the leaves	three kids sitting
1	three boys are jumping in the leaves	boys
	three kids are jumping in the leaves	kids
2	a man is jumping into an empty pool	an empty
	a man is jumping into a full pool	a full

Table 5: SICK data: Converting ORIG TO NONOVER

5.4 Textual Entailment

SemEval 2014 Task 1 (Marelli et al., 2014a) evaluates system predictions of textual entailment (TE) relations on sentence pairs from the SICK dataset (Marelli et al., 2014b). The three classes are entailment, contradiction and neutral. The sizes of SICK train, dev and test sets are 4439, 495 and 4906 pairs, respectively. We call this dataset ORIG.

We also create NONOVER, a copy of ORIG in which *the words that occur in both sentences have been removed.* A sentence in NONOVER is denoted by the special token <empty> if all words are removed. Table 5 shows three pairs from ORIG and their transformation in NONOVER. We observe

[7] For better comparability of approaches in our experiments, we use a simple SVM classifier, which performs slightly worse than Madnani et al. (2012)'s more complex meta-classifier.

[8] If we run ABCNN-3 (two conv) without the 15+3 "linguistic" features (i.e., MT and ROUGE), performance is 75.1/82.7.

that focusing on the non-overlapping parts provides clearer hints for TE than ORIG. In this task, we run two copies of each network, one for ORIG, one for NONOVER; these two networks have a single common LR layer.

Following Lai and Hockenmaier (2014), we train our final system (after fixing of hyperparameters) on train and dev (4,934 pairs). Our evaluation measure is accuracy.

5.4.1 Task-Specific Setup

We found that for this task forwarding two similarity scores from each block (instead of just one) is helpful. We use cosine similarity and Euclidean distance. As for paraphrase identification, we add the 15 MT features for each sentence pair based on the observation that entailed sentences are more likely to be paraphrases than contradictory sentences.

We use the following linguistic features.

Negation. Negation obviously is an important feature for detecting contradiction. Feature NEG is set to 1 if either sentence contains "no", "not", "nobody", "isn't" and to 0 otherwise.

Nyms. Following Lai and Hockenmaier (2014), we use WordNet to detect synonyms, hypernyms and antonyms in the pairs. But we do this on NONOVER (not on ORIG) to focus on what is critical for TE. Specifically, feature SYN is the number of word pairs in s_0 and s_1 that are synonyms. HYP0 (resp. HYP1) is the number of words in s_0 (resp. s_1) that have a hypernym in s_1 (resp. s_0). In addition, we collect all *potential antonym pairs* (PAP) in NONOVER. We identify the matched chunks that occur in *contradictory* and *neutral*, but not in *entailed* pairs. We exclude synonyms and hypernyms and apply a frequency filter of $n = 2$. In contrast to (Lai and Hockenmaier, 2014), we constrain the PAP pairs to cosine similarity above 0.4 in word2vec embedding space as this discards many noise pairs. Feature ANT is the number of matched PAP antonyms in a sentence pair.

Length. As before we use sentence length, both ORIG – LEN0O and LEN1O – and NONOVER lengths: LEN0N and LEN1N.

On the whole, we have 24 extra features: 15 MT metrics, NEG, SYN, HYP0, HYP1, ANT, LEN0O, LEN1O, LEN0N and LEN1N.

5.4.2 Results

Table 6 shows that our CNNs outperform the top three systems of SemEval. This demonstrates the promise of using deep learning for TE. Comparing ABCNN with BCNN, attention mechanism consistently improves performance. ABCNN-1 roughly has comparable performance as ABCNN-2 while ABCNN-3 has bigger improvement: a boost of 1.6 points.[9]

method		acc
SemEval Top3	(Jimenez et al., 2014)	83.1
	(Zhao et al., 2014)	83.6
	(Lai and Hockenmaier, 2014)	84.6
BCNN	one-conv	84.8*
	two-conv	85.0*
ABCNN-1	one-conv	85.6*
	two-conv	85.8*
ABCNN-2	one-conv	85.7*
	two-conv	85.8*
ABCNN-3	one-conv	86.0*
	two-conv	**86.2***

Table 6: Results on SICK. Significant improvements over (Lai and Hockenmaier, 2014) are marked with $*$ (McNemar's Chi-squared test, $p < .05$).

5.5 Visual Analysis

In Figure 4, we visualize the attention matrices for one TE sentence pair in ABCNN-2 for blocks b_1 (unigrams), b_2 (first convolutional layer) and b_3 (second convolutional layer). Darker shades of blue indicate stronger attention values.

In the left panel of Figure 4 (unigrams), each word corresponds to exactly one row or column. We can see that words in s_i with semantic equivalents in s_{1-i} get high attention while words without semantic equivalents get low attention, e.g., "walking" and "murals" in s_0 and "front" and "colorful" in s_1. This behavior seems reasonable for the unigram level.

Rows/columns of the middle attention matrix in Figure 4 correspond to phrases of length three since filter width $w = 3$. High attention values generally correlate with close semantic correspondence: the phrase "people are" in s_0 matches "several people are" in s_1; both "are walking outside" and "walking outside the" in s_0 match "are in front" in s_1; "the

[9]If we run ABCNN-3 (two conv) without the 24 linguistic features, the performance is 84.63.

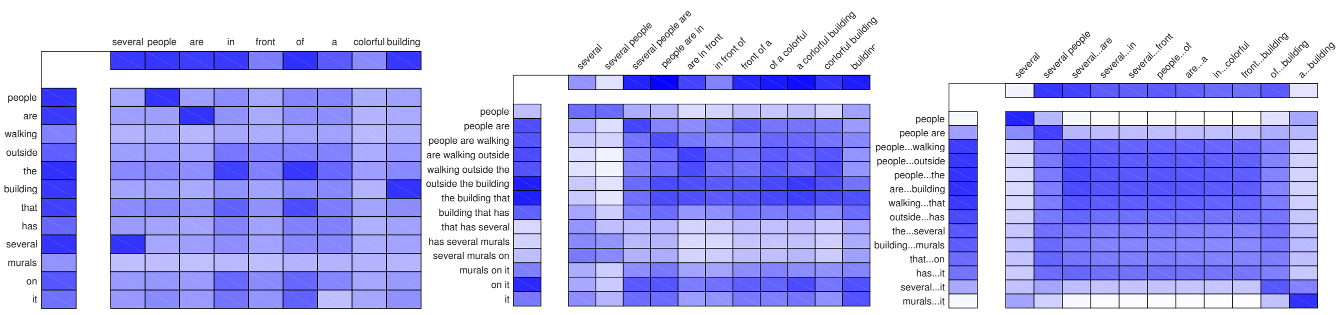


Figure 4: Attention visualization for TE: b_1 (unigrams, left), b_2 (conv1, center) and b_3 (conv2, right)

building that" in s_0 matches "a colorful building" in s_1. More interestingly, looking at the bottom right corner, both "on it" and "it" in s_0 match "building" in s_1; this indicates that ABCNN is able to detect some coreference across sentences. "building" in s_1 has two places in which higher attentions appear, one is with "it" in s_0, the other is with "the building that" in s_0. This may indicate that ABCNN recognizes that "building" in s_1 and "the building that" / "it" in s_0 refer to the same object. Hence, coreference resolution across sentences as well as within a sentence both are detected. For the attention vectors on the left and the top, we can see that attention has focused on the key parts: "people are walking outside the building that" in s_0, "several people are in" and "of a colorful building" in s_1.

Rows/columns of the rightmost attention matrix in Figure 4 (second layer of convolution) correspond to phrases of length 5 since filter width $w = 3$ in both convolution layers ($5 = 1 + 2 * (3 - 1)$). We use "..." to denote words in the middle if a phrase like "several...front" has more than two words. We can see that attention distribution in the matrix has focused on some local regions. As granularity of phrases is larger, it makes sense that the attention values are smoother. But we still can find some interesting clues: at the two ends of the main diagonal, higher attentions hint that the first part of s_0 matches well with the first part of s_1; "several murals on it" in s_0 matches well with "of a colorful building" in s_1, which satisfies the intuition that these two phrases are crucial for making a decision on TE in this case. This again shows the potential strength of our system in figuring out which parts of the two sentences refer to the same object. In addition, in the central part of the matrix, we can see that the long phrase "people are walking outside the building" in s_0 matches well with the long phrase "are in front of a colorful building" in s_1.

6 Summary

In this work, we presented three mechanisms to integrate attention into convolutional neural networks for general sentence pair modeling tasks. Our experimental results on AS, PI and TE show that attention-based CNNs perform better than CNNs without attention mechanisms. ABCNN-2 generally outperforms ABCNN-1 and ABCNN-3 surpasses both. In all tasks, we did not find any big improvement of two layers of convolution over one layer. This is probably due to the limited size of training data. We expect that, as larger training sets become available, deep ABCNNs will show even better performance.

Prior work on attention in deep learning mostly addresses LSTMs. LSTMs learn a sentence representation and use it for attention-based systems because they directly model next-word prediction while memorizing the entire sentence at the same time. But it is not clear whether this is the best strategy; e.g., in the AS example in Figure 1, it is possible to determine that "how much" in s_0 matches "\$161.5 million" in s_1 without taking the entire remaining sentence contexts into account. Our results suggest that CNNs benefit from incorporating attention into representations of local phrases detected by filters; and that encoding the whole context to form attention-based local representations (as LSTMs do) is unnecessary. Hence, attention is as useful for CNNs as it is for LSTMs.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of CVSC workshop*, pages 12–21.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of COLING*, pages 350–356.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *Proceedings of IEEE ASRU workshop*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of EMNLP*, pages 1576–1586.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of EMNLP*, pages 891–896.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. *SemEval*, pages 732–742.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *SemEval*, pages 329–334.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*, pages 1188–1196.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL text summarization workshop*, volume 8.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of NAACL*, pages 182–190.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval*, pages 1–8.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2015. A deep architecture for semantic matching with multiple positional sentence representations. *arXiv preprint arXiv:1511.08277*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*, pages 1744–1753.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of NAACL*, pages 901–911, May–June.

Wenpeng Yin and Hinrich Schütze. 2015b. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of ACL-IJCNLP*, pages 63–73.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS deep learning workshop*.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *SemEval*, pages 271–277.